M2T1L1_One_Hot_Encoding_rev

This week,
introduce to a different texture presentations.
explain how to convert text data into a numerical format using different methods: One-hot encoding, bag-of-words and TF-IDF
present documents in numerical format and understand the advantages and disadvantages of each method

1. **Why** Numerical Text Representation?
   - language can be ambiguous in assumes knowledge of contexts, representation of texts is one of the fundamental problems in text mining and information retrieval.
   - The goal of NLP is to be able to design algorithms to allow computers to "understand" natural language in order to perform some task
   - Computers are good with numbers

   - **How** do we numerically represent unstructured text documents to make them mathematically computable for different algorithms?
     - we represent every word with **a vector of 0s with a value of 1 at the position** at which the word appears in the vocabulary
       If we consider our **corpus** to be, this is a simple sentence with **five words**. Each word will be represented by **a vector of size five**. With each word is represented as a vector, our **sentence** is **a matrix of five-by-five**.

   - From Word to Document Representation
     - Our sentence is thus represented as:
       "This is a simple sentence" -> [[1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0], [0,0,0,1,0],[0,0,0,0,1]]

2. **One-Hot Encoder**
   - We show in this snippet and example of generating **one-hot encoding** for our example sentence using scikit-learn.
   - More formally if we take a corpus with **a vocabulary of size d** with **V, the number of unique words** in our vocabulary, then **every word w** is represented with a vector of size d with all 0s, but 1 at index i, corresponding to the index of the word w.
     - In a corpus with a vocabulary V of size d (number of unique words or dimensions), a word w is represented as a vector X of size d such as:

$$X_i^w = \begin{cases} 1, & \text{if } idx(w) = i \\ 0, & \text{otherwise} \end{cases}$$

   - **document**, which is a sequence of words, can be represented as **a matrix of size n by d**, with **n** being the **number of words in the document**, or **a single vector of dimension, d** with multiple values of 1 a where the words from the vocabulary are present.
     - Document: D = $this\ is\ a\ sentence$
     - Vocabulary: V= [$aardvak$,...,$this$,...,is,...,a,...,$sentence$,...,$zyther$]
     - One Hot Encoding: XD = [0,...,1,...,1,...,1,...,1,...,0]

3. **advantages and disadvantages** of One-Hot Encoding
   - main advantages:
     - simple and easy implementation.
   - However,
     - every word is represented as a vector of size of the vocabulary, this is not scalable for large vocabulary, think e.g., 100,000 words.
     - High dimensional and sparse matrices can be memory and computationally expensive
     - **all the word vectors are orthogonal**, there is no notion of similarity in meaning encoded in the vector representations.
     - The dot-product awards with similar meaning is the same as the dot-product of words with opposite meanings.

       E.g., the dot-product of good and great is 0 despite the similarity of meaning and equals to the dot-product of good and bad, which carry opposite meanings.

$$\left(W^{good}\right)^T . W^{great} = \left(W^{good}\right)^T . W^{bad} = 0$$

Summary

• Use One-Hot Encoding as a discrete text representation method to represent words and documents
• Understand the advantages and disadvantages of this method

M2T2L1_BoW

In this class, we will learn another effective discrete text presentation; Bag of Words. We will explain how to convert text data into numerical format using Bag of Words. By the end of the lecture, we will understand the advantages and disadvantages of using it.

1. **Bag of Words model** - represent each document as a **collection of word counts** while ignoring the sequence order in for simplicity
   - Represent a document as a column vector X of word counts
   - The bag of words is a fixed-length representation, which consists of a vector of word counts:
     Document: D, Vocabulary:, Bag of Words: X
   - The size of X is 1xd where d is the size of the vocabulary.
   - A **collection** of n documents is represented with a matrix of size n x d
     - **example** of a movie review: Using Bag of Words model, we count the occurrence of every word in the document.
       (For example, the word "it" is present six times in the review, "I" five times, and so on)
       every document is represented as a vector X of word counts
       size of the document vector = vocabulary size D, where the values are equal to the number of times a word occurred.
     - example - "this was the best of times", "this was the worst of times"
       vector x contains 0 where the word from the vocabulary is not present, two for the word "it" that appears twice, etc.

2. **how to create** the text presentation of an example sentence using Bag of Words model
   - use the '**CountVectorizer'** from '**Sklearn' library**.
     - X is the vector representation of my text and contains the frequency counts of each word present in my text.
     - The second output shows a list of words present.

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 my_text=["it was the best of times, it was the worst of times"]
3 vectorizer = CountVectorizer()
4 X = vectorizer.fit_transform(my_text)
5 print(X.toarray())
6 vectorizer.get_feature_names()
```

```
[[1 2 2 2 2 2 1]]
['best', 'it', 'of', 'the', 'times', 'was', 'worst']
```

3. **Advantages and Disadvantages** of Bag of Words
   - Advantages:
     - simple and easy to implement
   - Disadvantages:
     - every document is represented with a vector of size equal to the vocabulary size, which is not scalable considering the larger vocabulary and languages.
     - Similar to one-hot encoding, high dimensional sparse matrices can be memory and computationally expensive.
     - Bag of Words model is based on words frequency counts and doesn't take into consideration the order of words. Thus, the meaning from the context is not captured.

Summary
- Use Bag of Words as a discrete text representation method to represent documents
- Understand the advantages and disadvantages of this method

M2T3L1_TF_IDF

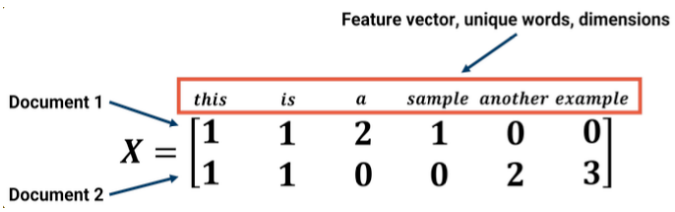
In this class, we will learn another effective discrete text representation, TF-IDF. We will explain how to convert text data into a numerical format using it. By the end of the lecture, we will understand the advantages and disadvantages of using TF-IDF and also the superiority of this method over bag of words or one-hot encoded.

1. **Why Do We Need TF-IDF?**
   - In our previous lecture, we found out the bag of words approach does not emphasize on the importance of each word.

     For example, we have this sentence: This is the NLP class.
   - If you were to use a method such as bag of words, it would have given us the same importance for every word in the sentence, for example, the is as important as NLP word in our example sentence.
   - **TF-IDF** algorithm will help us assign a **more logical importance** to a vector of words given the documents.

   - **What is TF-IDF and when to use it?**
     - TF-IDF in short means a word's importance score in a document among N number of documents.
     - If you are going to convert documents into word count, you would like to use TF-IDF method.

   - **Simple example**
     - A vocabulary vector for all the words in **N** documents with **six unique words** or terms: this, is, a, sample, another, example.



- Let's assume **two documents** contained different sentence.
- **bag of words approach** needs to **create** our **word count matrix** or **document term matrix**, **X**. The column of this matrix has the same size as our **vocabulary vector**, **V**.
- **create term and term count table** for each document. Each row of document term matrix X includes a document.
  For the first document, 'this', 'is' and 'sample', 1 time in the Document 1, have a value of 1, and 'a' appear 2 times in Document 1 and 'another' and 'example' appear 0 time in Document 1. use the same approach for the second document

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

⇨ 'a', 'another', and 'example' have the highest values among others, but they may not be the best representative of their documents.

2. how TF-IDF can help us differentiate important words comparing to others? TF-IDF has two parts.

- **TF**: term frequency
  - calculate the term frequency for every single unique word in our vocabulary → a ratio that reflects the total number of appearance of unique words or terms in a document.
  - if a term appears many times in a document, term frequency will be a high value.
    'this' appeared 1 time in document 1, the total number of term count in this document = 5.

$$\text{tf}(''\text{this}'', d_1) = \frac{1}{5} = 0.2$$
$$\text{tf}(''\text{this}'', d_2) = \frac{1}{7} \approx 0.14$$

  - term frequency = number of times a term appear in the document / total number of term count in that document
    this has a TF value of 0.2 for Document 1 and has a TF value of 0.14 for Document 2.
  - Term frequency on its own cannot be a good representative of a word's importance, for example, common words like a can have high term frequency values, which are not the key representative for documents → need to use the **IDF**.

- **IDF:** inverse document frequency
  - N is the total number of documents in the corpus. In this example, N=2.
    This value needs to be divided down by the number of documents containing each unique words or term.
    For example, the word this appeared in two documents. Therefore, the IDF value of this document is equal to log of 1, which is equal to 0. Common words like a, the, and this will have a very low value of IDF in general.

$$= \log(\frac{N}{\textit{The number of documents containing that } \boldsymbol{term}})$$

$$\text{idf}(''\text{this}'', D) = \log\left(\frac{2}{2}\right) = 0$$

  - IDF will help us **reduce the effects of term frequency for common words**.
    For example, if we were to recreate the return document matrix X using TF-IDF approach the word this will have 0 value in matrix X instead of 1.

- **TF-IDF:** A word's importance score in a document, among N documents
  Final score = TF * IDF → A higher scoring TF-IDF shows a more characteristic of each term in a document.

$$\text{tfidf}(''\text{this}'', d_1, D) = 0.2 \times 0 = 0$$
$$\text{tfidf}(''\text{this}'', d_2, D) = 0.14 \times 0 = 0$$

3. **Advantages and Disadvantages of TF-IDF**
   - Advantages:
     - Simple and easy to implement
     - Higher score means “more characteristic”. Common words will have very small scores such as “the”, “a”, “this”,...
     - TF-IDF is a good technique to search for document, find similar documents, or cluster documents
   - Disadvantages:
     - TF-IDF does NOT consider the position of the words because of creating the document-term matrix. Other methods such as Bag of Words also suffers from this issue

Summary

• Use TF-IDF as a better alternative for discrete text representation comparing to one-hot-encoding and bag of words

• Understand the advantages and disadvantages of this method